UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/728,370	12/03/2003	Glen Darling	50325-0841	6391

29989 7590 09/25/2006

HICKMAN PALERMO TRUONG & BECKER, LLP
2055 GATEWAY PLACE
SUITE 550
SAN JOSE, CA 95110

EXAMINER	
STEELMAN, MARY J	

ART UNIT	PAPER NUMBER
2191	

DATE MAILED: 09/25/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	10/728,370	DARLING ET AL.	
	Examiner	Art Unit	
	Mary J. Steelman	2191	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *09 June 2006*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-31* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-31* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This Office Action is in response to RCE, Remarks and Amendments received 9 June 2006. Per Applicant's request, claims 1, 2, 6, 7, 8, 13, 14, 15, 20, 21, and 22 have been amended. New claims 27-31 have been added. Claims 1-31 are pending.

Specification

2. Applicant is requested to amend page 1 of the Specification to include the serial number of the related application.

Claim Objections

3. Claim 1 recites "process of DLL" in lines 14 and 17. Should be –process or DLL-- Change 'of' to 'or'.

Claim 6 recites "process of DLL" in lines 14 and 17. Should be –process or DLL-- Change 'of' to 'or'.

Claim 6 lacks a period ('.') at the end. Delete ';' and enter '.'

Claim 13 recites "process of DLL" in lines 12 and 15. Should be –process or DLL-- Change 'of' to 'or'.

Claim 13 lacks a period ('.') at the end. Delete ';' and enter '.'

Claim 20 recites "process of DLL" in lines 14 and 17. Should be –process or DLL-- Change 'of' to 'or'.

Claim 20 lacks a period ('.') at the end. Delete ';' and enter '.'

Claim 27 recites "process of DLL" in lines 19 and 23. Should be –process or DLL-- Change 'of' to 'or'.

Claim 27 appears to be missing (a) through (d) indicators.

Claim 29 recites, "A method as recited in Claim 27...", should be –An apparatus as recited in claim 27...-- Delete 'A method' and insert 'An apparatus'.

Claim Rejections - 35 USC § 101

4. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

Claims 20-26 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter. It appears to Examiner that 'a computer-readable medium', as defined in the Specification, page 58, intends to include 'carrier waves'. This is a non-statutory embodiment. Claim may be cured by amending to recite, "A computer-readable medium, tangibly embodied on a physical storage medium, carrying one or more sequences..."

Response to Arguments

5. Applicant's arguments with respect to claims 1, 6, 13, 20, and 27 have been considered but are moot in view of the new grounds of rejection.

Claim Rejections - 35 USC § 103

6. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

7. Claims 1-31 are rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent 5,754,858 to Broman et al., in view of US Patent Application Publication 2003/0217193 A1 to Thurston et al., and further in view of US Patent Application Publication 2004/0133875 A1 to Kramer.

Per claim 1:

A method of a development and build environment for packaged software delivery in a distributed network of nodes, the method comprising the computer-implemented steps of:

Broman disclosed a 'method' (col. 23, line 31-col. 24, line 40) for "customizable automated application project generation (col. 4, lines 65-66) (development and build environment). Broman disclosed the 'network' suitability at col. 20, lines 52-63.

-compiling source code files into executable file modules;

Broman disclosed compiling source code at col. 6, lines 20-22, "The writer builds...from the source files...using a compiler..."

-wherein a module contains an image for a process or a dynamically linked library (DLL);

Broman disclosed dynamically linked libraries at col. 6, lines 27-32, "...links the object code...and .RES files together into a dynamic link library..."

-creating a software package that comprises at least one module;

Broman disclosed (Col. 6, lines 36-37) the custom application project generator creating the application project (software package comprising at least one module).

-wherein packages are created based on features/characteristics or purpose;

Broman disclosed, as an example, (col. 8, lines 5-10) the package may be language specific (feature / characteristic / purpose).

-creating metadata for a module that includes any module information such as the module's: binary signature, name, directory path, and characteristics;

Broman disclosed (col. 7, line 55), as an example, a 'name' may be created. Also, col. 11, lines 47-54, disclosed a services module generates a file named by destination-filename (name) which can contain a path (directory path).

-inserting the module's metadata into the software package;

Broman disclosed a 'newproj.inf' template (col. 11, line 39) that contains created directories, whereby the directories are filled by the services module (col. 11, line 49). The services module lists files used to build the application (col. 11, lines 60-67).

Broman failed to explicitly disclose:

-gathering application program interface (API) dependency information for a first module, wherein the first module can provide and use at least one API, by

(a) receiving a list of dependent modules for a given process or DLL module;

(b) storing, in the first module's metadata, dependency information for the dependent modules in the list wherein dependency information includes API names and version that the process or DLL depends on;

However, Thurston, more specifically disclosed metadata [0008-0009] to control installations and dependency information placed into 'constraints' that must be satisfied into a header. See FIG. 4. Thurston disclosed 'metadata' held in header files [0039-0041], including version number, names of dependent devices, size and checksums, and digital signature.

Broman / Thurston failed to disclose:

(c) collecting additional dependency information documented in one or more additional modules specifications wherein, dependency information includes API and versions that the process or DLL depends on;

(d) storing the additional dependency information in the first module's metadata.

However, Brodkorb disclosed [0025-0026] a software deliver archive (SDA), a ZIP and JAR compatible archive format. Manifests contain context and dependency information. [0027], SDAs can be nested, with the manifest information being passed on to the SDA at the highest level (storing the additional dependency information in the first module's metadata). [0023-0024], A software delivery manager (SDM) manages and deploys software delivery archives (SDAs). See FIG. 2 with nested SDA's, #60.

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention to modify Broman's invention to include additional details in the component header such as names, versions, and dependency constraints, as disclosed by Thurston, because such information contained in a header file as meta data is useful for conveying relevant information when delivering software for installation. Thurston was motivated (Thurston –[0010]) to provide updates to support new and different types of devices in a manner to simplify development and maintenance. It would have been obvious, to one of ordinary skill in the art, at the time of the invention to further modify Broman / Thurston, using Brodkorb, because Brodkorb provided support for software development [0007], understanding the need for [0004] upgrades, migrations, newer software releases. Brodkorb felt the need to manage dependencies [0017] when software updates are deployed.

Per claim 2:

-a linker creates the list of dependent modules for a given process or DLL module and places the list....

Broman disclosed a linker. See FIG. 2, #72 & #88. Broman disclosed (col. 8, lines 5-10 & 25-28) selectable option steps and selectable language resources, disclosing consideration to dependencies. The services module copies (col. 11, lines 55-60) template information to the project. Col. 11, lines 63-67, disclosed a 'make file' which lists source code files that must be compiled to build (list of dependent modules for a given process) the application.

Broman disclosed 'dependency' information, and failed to disclose placing the list in the first module's metadata.

However, Brodkorb disclosed [0025-0026] a software deliver archive (SDA), a ZIP and JAR compatible archive format. Manifests contain context and dependency information. [0027], SDAs can be nested, with the manifest information being passed on to the SDA at the highest level (storing the additional dependency information in the first module's metadata). [0023-0024], A software delivery manager (SDM) manages and deploys software delivery archives (SDAs). See FIG. 2 with nested SDA's, #60.

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention to modify Broman's invention to include additional details in the component header such as names, versions, and dependency constraints, as disclosed by Thurston, because such information contained in a header file as meta data is useful for conveying relevant information when delivering software for installation. Thurston was motivated (Thurston –[0010]) to provide updates to support new and different types of devices in a manner to simplify development and maintenance. It would have been obvious, to one of ordinary skill in the art, at the time of the invention to further modify Broman / Thurston, using Brodkorb, because Brodkorb provided support for software development [0007], understanding the need for [0004] upgrades,

migrations, newer software releases. Brodkorb felt the need to manage dependencies [0017] when software updates are deployed.

Per claim 3:

-creating metadata for each API;

Broman disclosed (col. 9, lines 20-55) that a user is presented with options for creating an application project. The project generator and services module generate the application project using the template files / resource files. Template files / resource files contain text and binary files and macros and directives, which determine the content of the source files in the application project (provides metadata type information). Broman disclosed (col. 10, lines 3-12 & 31-33) 'directives' guide the production of the application project, through a confirm.inf template.

-inserting the API metadata into the software package;

Broman disclosed (col. 11, lines 17-21) "The newproj.inf template contain the instructions (metadata type information inserted into package) that the services module uses to construct the application project. The instructions are statements, directives, and macros that work together to describe the structure of the application project."

Regarding the limitation:

-wherein metadata for an API includes, but is not limited to: the API's name and version.

Broman was suggestive of metadata included in the created application project, but failed to explicitly disclose metadata and contents. However Thurston disclosed metadata stored in a

header portion [0008-0009] to be used to control installation of code. Thurston disclosed the header data could include a name, version, size, checksum, digital signature.

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention to modify Broman's invention to include additional details of metadata in the component header, as disclosed by Thurston, because such information contained in a header file as metadata is useful for conveying relevant information when delivering software for installation. Thurston was motivated (Thurston –[0010]) to provide updates to support new and different types of devices in a manner to simplify development and maintenance. It would have been obvious, to one of ordinary skill in the art, at the time of the invention to further modify Broman / Thurston, using Brodkorb, because Brodkorb provided support for software development [0007], understanding the need for [0004] upgrades, migrations, newer software releases. Brodkorb felt the need to manage dependencies [0017] when software updates are deployed.

Per claim 4:

Broman failed to disclose:

-calculating a binary signature for each module and inserting the binary signature into the respective module's metadata;

-wherein each unique version of a module will have a unique binary signature.

However Thurston disclosed metadata stored in a header portion [0008-0009] to be used to control installation of code. Thurston disclosed the header data could include a name, version, size, checksum, digital signature (binary signature). Details including name, version, size, checksum contribute to a unique binary signature.

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention to modify Broman's invention to include additional details of metadata in the component header, as disclosed by Thurston, because such information contained in a header file as metadata is useful for conveying relevant information when delivering software for installation. Thurston was motivated (Thurston –[0010]) to provide updates to support new and different types of devices in a manner to simplify development and maintenance. It would have been obvious, to one of ordinary skill in the art, at the time of the invention to further modify Broman / Thurston, using Brodkorb, because Brodkorb provided support for software development [0007], understanding the need for [0004] upgrades, migrations, newer software releases. Brodkorb felt the need to manage dependencies [0017] when software updates are deployed.

Per claim 5:

Broman failed to specifically disclose:

-creating metadata for a package that includes any package information such as the package's: name, build date, and characteristics;

-inserting the package metadata into the package.

However Thurston disclosed metadata stored in a header portion (inserting package metadata) [0008-0009] to be used to control installation of code. Thurston disclosed the header data could include a name, version, size, checksum, digital signature (binary signature). Details including name, version (build date), size, checksum (characteristics) contribute to a unique identity.

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention to modify Broman's invention to include additional details of metadata in the component header, as disclosed by Thurston, because such information contained in a header file as metadata is useful for conveying relevant information when delivering software for installation. Thurston was motivated (Thurston –[0010]) to provide updates to support new and different types of devices in a manner to simplify development and maintenance. It would have been obvious, to one of ordinary skill in the art, at the time of the invention to further modify Broman / Thurston, using Brodkorb, because Brodkorb provided support for software development [0007], understanding the need for [0004] upgrades, migrations, newer software releases. Brodkorb felt the need to manage dependencies [0017] when software updates are deployed.

Per claim 6:

A method of a development and build environment for packaged software delivery in a distributed network of nodes, the method comprising the computer-implemented steps of:

-compiling source code files into executable file modules;

-wherein a module contains an image for a process or a dynamically linked library (DLL);

-creating a software package that comprises at least one module;

-creating metadata for a module that includes any module information such as

the module's: binary signature, name, directory path, and characteristics;

-inserting the module's metadata into the software package;

-gathering application program interface (API) dependency information for the first module

wherein the first module can provide and use at least one API, by;

(a) receiving a list of dependent modules for a given process of DLL module;

(b) storing, in the first module's metadata, dependency information for the dependent modules in the list wherein dependency information includes API names and version that the process of DLL depends on;

See rejection of limitations as addressed in claim 1 above.

Per claim 7:

-wherein a linker creates the list of dependent modules for a given process or DLL module and places the list in the first module's metadata.

See rejection of limitations as addressed in claim 2 above.

Per claim 8:

-collecting additional dependencies documented in one or more additional module specifications and placing them into the first module's metadata;

-wherein the dependencies documented in each module lists API names and versions that the process or DLL depends on.

See rejection of limitations as addressed in claim 1 above.

Per claim 9:

-creating metadata for each API;

-inserting the API metadata into the software package;

-wherein metadata for an API includes, but is not limited to: the API's name and version.

See rejection of limitations as addressed in claim 3 above.

Per claim 10:

-calculating a binary signature for each module and inserting the binary signature into the respective module's metadata;

-wherein each unique version of a module will have a unique binary signature.

See rejection of limitations as addressed in claim 4 above.

Per claim 11:

-packages are created based on features/characteristics or purpose.

See rejection of limitations as addressed in claim 1 above.

Per claim 12:

-creating metadata for a package that includes any package information such as the package's: name, build date, and characteristics;

-inserting the package metadata into the package.

See rejection of limitations as addressed in claim 5 above.

Per claim 13:

An apparatus for a development and build environment for packaged software delivery in a distributed network of nodes, comprising:

Bronson disclosed a 'system' (apparatus) (col. 4, lines 64-66) providing an automated application project generation (a development and build environment).

-means for compiling source code files into executable file modules;

-wherein a module contains an image for a process or a dynamically linked library (DLL);

-means for creating a software package that comprises at least one module;

-means for creating metadata for a module that includes any module information such as the module's: binary signature, name, directory path, and characteristics;

-means for inserting the module's metadata into the software package;

-means for gathering application program interface (API) dependency information for the first module wherein the first module can provide and use at least one API by

(a) receiving a list of dependent modules for a given process of DLL module;

(b) storing, in the first module's metadata, dependency information for the dependent modules in the list wherein dependency information includes API names and version that the process of DLL depends on;

See rejection of limitations as addressed in claim 1 above.

Per claim 14:

-a linker;

-wherein said linker creates a list of dependent modules for a given process or DLL module and places the list in the first module's metadata.

See rejection of limitations as addressed in claim 2 above.

Per claim 15:

-means for collecting additional dependencies documented in one or more additional module specifications and placing them into the first module's metadata;

-wherein the dependencies documented in each module lists API names and versions that the process or DLL depends on.

See rejection of limitations as addressed in claim 1 above.

Per claim 16:

-means for creating metadata for each API;

-means for inserting the API metadata into the software package;

-wherein metadata for an API includes, but is not limited to: the API's name and version.

See rejection of limitations as addressed in claim 3 above.

Per claim 17:

-means for calculating a binary signature for each module and inserting the binary signature into the respective module's metadata;

-wherein each unique version of a module will have a unique binary signature.

See rejection of limitations as addressed in claim 4 above.

Per claim 18:

-packages are created based on features/characteristics or purpose.

See rejection of limitations as addressed in claim 1 above.

Per claim 19:

-means for creating metadata for a package that includes any package information such as the package's: name, build date, and characteristics;

-means for inserting the package metadata into the package.

See rejection of limitations as addressed in claim 5 above.

Per claim 20:

A computer-readable medium carrying one or more sequences of instructions for a development and build environment for packaged software delivery in a distributed network of nodes, which

instructions, when executed by one or more processors, cause the one or more processors to carry out the steps of:

-compiling source code files into executable file modules;

-wherein a module contains an image for a process or a dynamically linked library (DLL);

-creating a software package that comprises at least one module;

-creating metadata for a first module that includes any module information such as the module's: binary signature, name, directory path, and characteristics;

-inserting the module's metadata into the software package;

-gathering application program interface (API) dependency information for the first module wherein the first module can provide and use at least one API by

(a) receiving a list of dependent modules for a given process of DLL module;

(b) storing, in the first module's metadata, dependency information for the dependent modules in the list wherein dependency information includes API names and version that the process of DLL depends on;

This is a 'computer readable medium' version of claim 1. See FIG. 2 where the instructions for development are shown located on a system. See rejection of limitations as addressed in claim 1 above.

Per claim 21:

-wherein a linker creates the list of dependent modules for a given process or DLL module and places the list in the first module's metadata.

See rejection of limitations as addressed in claim 2 above.

Per claim 22:

-collecting additional dependencies documented in one or more additional module specifications and placing them into the first module's metadata;

-wherein the dependencies documented in each module lists API names and versions that the process or DLL depends on.

See rejection of limitations as addressed in claim 1 above.

Per claim 23:

-creating metadata for each API;

-inserting the API metadata into the software package;

-wherein metadata for an API includes, but is not limited to: the API's name and version.

See rejection of limitations as addressed in claim 3 above.

Per claim 24:

-calculating a binary signature for each module and inserting the binary signature into the respective module's metadata;

-wherein each unique version of a module will have a unique binary signature.

See rejection of limitations as addressed in claim 4 above.

Per claim 25:

-packages are created based on features/characteristics or purpose.

See rejection of limitations as addressed in claim 1 above.

Per claim 26:

-creating metadata for a package that includes any package information such as the package's: name, build date, and characteristics; and inserting the package metadata into the package.

See rejection of limitations as addressed in claim 5 above.

Per claim 27:

An apparatus running a development and build environment for packages software delivery in a distributed network of nodes, the apparatus comprising:

-a network interface that is coupled to a data network for receiving one or more packet flows therefrom; (Broman, col. 4, line 33)

-a processor; (Broman, col. 4, line 2)

-one or more stored sequences of instructions which, when executed by the processor, cause the processor to carry out the steps of:

-compiling source code files into executable file modules;

wherein a module contains an image for a process or a dynamically linked library (DLL);

creating a software package that comprises at least one module;

wherein packages are created based on features/characteristics or purpose;

creating metadata for a first module that includes any module information such as the module's: binary signature, name, directory path, and characteristics;

inserting the module's metadata into the software package;

gathering application program interface (API) dependency information for the first module wherein the first module can provide and use at least one API, by

(e) receiving a list of dependent modules for a given process of DLL module;

(f) storing, in the first module's metadata, dependency information for the dependent modules in the list wherein dependency information includes API names and version that the process of DLL depends on;

(g) collecting additional dependency information documented in one or more additional modules specifications wherein, dependency information includes API and versions that the process or DLL depends on;

(h) storing the additional dependency information in the first module's metadata.

See rejection of limitations addressed in claim 1 above.

Per claim 28:

-wherein a linker creates the list of dependent modules for a given process or DLL module and places the list in the first module's metadata.

See rejection of limitations addressed in claim 2 above.

Per claim 29:

creating metadata for each API;

inserting the API metadata into the software package;

wherein metadata for an API includes, but is not limited to: the API's name and version.

See rejection of limitations addressed in claim 3 above.

Per claim 30:

-calculating a binary signature for each module and inserting the binary signature into the respective module's metadata;

wherein each unique version of a module will have a unique binary signature.

See rejection of limitations addressed in claim 4 above.

Per claim 31:

-creating metadata for a package that includes any package information such as the package's: name, build date, and characteristics;

inserting the package metadata into the package.

See rejection of limitations addressed in claim 5 above.

Conclusion

8. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Mary Steelman, whose telephone number is (571) 272-3704. The examiner can normally be reached Monday through Thursday, from 7:00 AM to 5:30 PM If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Wei Zhen can be reached at (571) 272-3708. The fax phone number for the organization where this application or proceeding is assigned: 571-273-8300.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Mary Steelman 

08/31/2006